
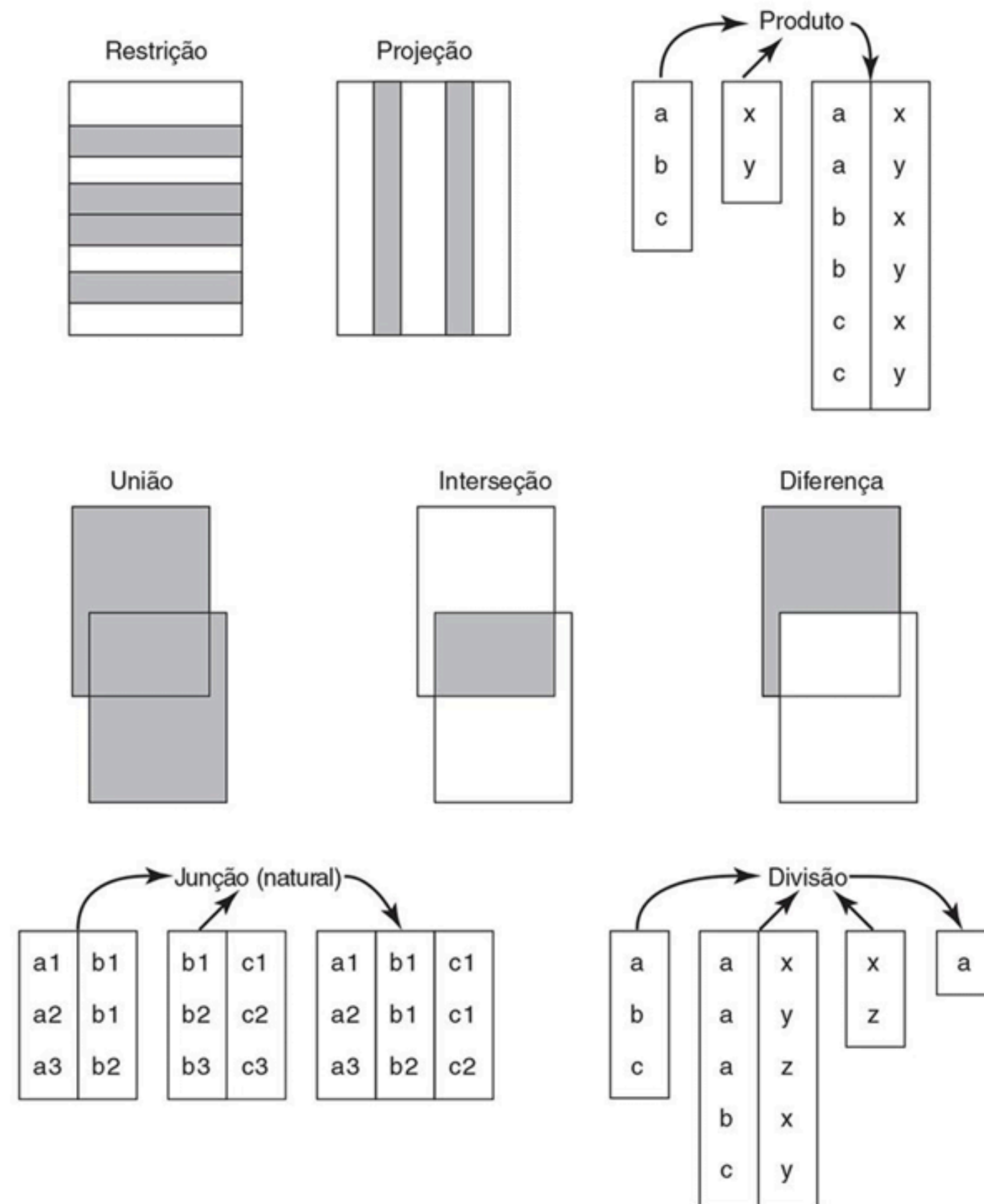


# Álgebra Relacional

 Adenilson Alves  
Luan Ferreira  
Victor Bonfim

# Tópicos

- **Introdução**
- **Revendo o fechamento**
- **A álgebra original: sintaxe**
- **A álgebra original: semântica**
- **Exemplos**
- **Para que serve a álgebra?**
- **Pontos adicionais**
- **Operadores adicionais**
- **Agrupamento e desagrupamento**
- **Resumo Exercícios**
- **Referências e bibliografia**



A álgebra relacional é uma coleção de operadores que tomam relações como seus operandos e retornam uma relação como seu resultado.

# Propriedade de Fechamento

## **Fechamento:**

A saída de uma operação relacional é outra relação.

## **Aninhamento:**

Permite o aninhamento de expressões relacionais, como ocorre com expressões aritméticas.

## **Problema:**

Relações derivadas precisam de cabeçalhos adequados para operações subsequentes.



# Exemplo

## Operador: **RENAME**

Renomeia atributos de uma relação, mantendo o restante inalterado.

F RENAME CIDADE AS FCIDADE

Essa expressão – observe que ela é uma expressão, e não um “comando” ou instrução, e portanto pode estar aninhada dentro de outras expressões – produz uma relação que tem o mesmo cabeçalho e corpo da relação que é o valor atual da variável de relação F, exceto pelo fato de o atributo de cidade ser chamado FCIDADE, em vez de CIDADE:

F#	FNOME	STATUS	FCIDADE
F1	Smith	20	Londres
F2	Jones	10	Paris
F3	Blake	30	Paris
F4	Clark	20	Londres
F5	Adams	30	Atenas



# Álgebra Original: Sintaxe

---

Sintaxe baseada no Tutorial D

Expressões de álgebra relacional: 8 operadores + RENAME

Inclui notas de semântica

Notação matemática vs. palavras-chave

Preferência por palavras-chave: JOIN, WHERE

Expressões mais longas mas compreensíveis

<i>Símbolo</i>	<i>Operação</i>	<i>Sintaxe</i>	<i>Tipo</i>
<b><math>\sigma</math></b>	Seleção / Restrição	$\sigma_{\text{condição}} ( \text{Relação} )$	Primitiva
<b><math>\pi</math></b>	Projeção	$\pi_{\text{expressões}} ( \text{Relação} )$	Primitiva
<b><math>\cup</math></b>	União	$\text{Relação1} \cup \text{Relação2}$	Primitiva
<b><math>\cap</math></b>	Intersecção	$\text{Relação1} \cap \text{Relação2}$	Adicional
<b>-</b>	Diferença de conjuntos	$\text{Relação1} - \text{Relação2}$	Primitiva
<b>x</b>	Produto cartesiano	$\text{Relação1} \times \text{Relação2}$	Primitiva
<b> x </b>	Junção	$\text{Relação1} \bowtie \text{Relação2}$	Adicional
<b><math>\div</math></b>	Divisão	$\text{Relação1} \div \text{Relação2}$	Adicional
<b><math>\rho</math></b>	Renomeação	$\rho_{\text{nome}} ( \text{Relação} )$	Primitiva
<b><math>\leftarrow</math></b>	Atribuição	$\text{variável} \leftarrow \text{Relação}$	Adicional



# Álgebra Original: Sintaxe

---

```
RELATION { <lista_com_vírgulas_de_expressão_de_tupla> }  
<nome_de_variável_de_relação>  
<invocação_de_operador_relacional>  
<expressão_with>  
<nome_introduzido>  
( <expressão_relacional> )
```





# Álgebra Original: Semântica

---

**União:** Combina tuplas de duas relações, Remove duplicatas, Exige mesmo número de colunas e tipos compatíveis

**Interseção:** Filtra elementos comuns, Exige estrutura compatível entre relações

**Divisão:** Mostra tuplas que estão em uma relação, mas não na outra, Exclui elementos correspondentes, Estrutura de ambas as relações deve ser igual

A

F#	FNOME	STATUS	CIDADE
F1	Smith	20	Londres
F4	Clark	20	Londres

B

F#	FNOME	STATUS	CIDADE
F1	Smith	20	Londres
F2	Jones	10	Paris

1. *União*  
(A UNION B)

F#	FNOME	STATUS	CIDADE
F1	Smith	20	Londres
F4	Clark	20	Londres
F2	Jones	10	Paris

2. *Interseção*  
(A INTERSECT B)

F#	FNOME	STATUS	CIDADE
F1	Smith	20	Londres

3. *Diferença*  
(A MINUS B)

F#	FNOME	STATUS	CIDADE
F4	Clark	20	Londres

4. *Diferença*  
(B MINUS A)

F#	FNOME	STATUS	CIDADE
F2	Jones	10	Paris

F WHERE CIDADE = 'Londres'

F#	FNOME	STATUS	CIDADE
F1	Smith	20	Londres
F4	Clark	20	Londres

P WHERE PESO <  
PESO (14.0)

P#	PNOME	COR	PESO	CIDADE
P1	Porca	Vermelho	12.0	Londres
P5	Came	Azul	12.0	Paris

FP WHERE F# = F# ('F6')  
OR P# = P# ('P7')

F#	P#	QDE

**Restrição:** Produto cartesiano: combina tuplas de duas relações, Gera nova relação com todas as colunas, Renomeação necessária para atributos iguais

A

F#
F1
F2
F3
F4
F5

B

P#
P1
P2
P3
P4
P5
P6

Produto cartesiano (A TIMES B)

F#	P#										
F1	P1	F2	P1	F3	P1	F4	P1	F5	P1		
F1	P2	F2	P2	F3	P2	F4	P2	F5	P2		
F1	P3	F2	P3	F3	P3	F4	P3	F5	P3		
F1	P4	F2	P4	F3	P4	F4	P4	F5	P4		
F1	P5	F2	P5	F3	P5	F4	P5	F5	P5		
F1	P6	F2	P6	F3	P6	F4	P6	F5	P6		
..	..	..	..	..	..	..	..				

**Produto Cartesiano:** Restrição: aplica condição sobre relação, Retorna nova relação com tuplas que atendem à condição, Filtra apenas tuplas específicas

F { CIDADE }	<table><tr><th>CIDADE</th></tr><tr><td>Londres</td></tr><tr><td>Paris</td></tr><tr><td>Atenas</td></tr></table>	CIDADE	Londres	Paris	Atenas						
CIDADE											
Londres											
Paris											
Atenas											
P { COR, CIDADE }	<table><tr><th>COR</th><th>CIDADE</th></tr><tr><td>Vermelho</td><td>Londres</td></tr><tr><td>Verde</td><td>Paris</td></tr><tr><td>Azul</td><td>Oslo</td></tr><tr><td>Azul</td><td>Paris</td></tr></table>	COR	CIDADE	Vermelho	Londres	Verde	Paris	Azul	Oslo	Azul	Paris
COR	CIDADE										
Vermelho	Londres										
Verde	Paris										
Azul	Oslo										
Azul	Paris										
(F WHERE CIDADE = 'Paris' ) { F# }	<table><tr><th>F#</th></tr><tr><td>F2</td></tr><tr><td>F3</td></tr></table>	F#	F2	F3							
F#											
F2											
F3											

**Projeção:** Projeção: seleciona atributos específicos, Gera nova relação com tuplas únicas,  
Elimina duplicatas

F#	FNOME	STATUS	CIDADE	P#	PNOME	COR	PESO
F1	Smith	20	Londres	P1	Porca	Vermelho	12.0
F1	Smith	20	Londres	P4	Parafuso	Vermelho	14.0
F1	Smith	20	Londres	P6	Tubo	Vermelho	19.0
F2	Jones	10	Paris	P2	Pino	Verde	17.0
F2	Jones	10	Paris	P5	Came	Azul	12.0
F3	Blake	30	Paris	P2	Pino	Verde	17.0
F3	Blake	30	Paris	P5	Came	Azul	12.0
F4	Clark	20	Londres	P1	Porca	Vermelho	12.0
F4	Clark	20	Londres	P4	Parafuso	Vermelho	14.0
F4	Clark	20	Londres	P6	Tubo	Vermelho	19.0

**Junção:** combina tuplas com valores iguais em atributos comuns, Gera nova relação com todos os atributos de A e B, Retorna tuplas que atendem à igualdade nos atributos correspondentes

DEND		MED			
	F#	F#	P#	F#	P#
	F1	F1	P1	F2	P1
	F2	F1	P2	F2	P2
	F3	F1	P3	F3	P2
	F4	F1	P4	F4	P2
	F5	F1	P5	F4	P4
		F1	P6	F4	P5
		..	..		
DOR		DOR		DOR	
	P#		P#		P#
	P1		P2		P1
			P4		P2
					P3
					P4
					P5
					P6
DEND DIVIDEBY DOR PER MED					
	F#		F#		F#
	F1		F1		F1
	F2		F4		

**Divisão:** seleciona tuplas associadas a todas as tuplas de outra relação, Retorna resultado com atributos específicos, Foco em tuplas que satisfazem todos os critérios

<https://github.com/luanferreiradev>



# Exemplos

Objetivo: Demonstrar o uso de expressões de álgebra relacional para consultas de banco de dados.

## Exemplo 1: União

```
<localhost> Script  *<localhost> Script-1 x
#use SistemaExemplo;
SELECT nomeCliente FROM Clientes WHERE cidade = 'São Paulo'
UNION
SELECT nomeCliente FROM Clientes WHERE cidade = 'Rio de Janeiro';
```

Resultados 1 X

SELECT nomeCliente FROM Clientes

Grade		A-Z nomeCliente
1		Ana
2		Carlos
3		Bruno
Texto		



## Exemplo 2: Interseção

The screenshot shows a SQL IDE with two tabs: "<localhost> Script" and "\*<localhost> Script-1 x". The active tab contains the following SQL query:

```
#use SistemaExemplo;

SELECT
    c.idCliente,
    c.nomeCliente,
    c.cidade,
    p.numeroPedido,
    p.dataPedido
FROM
    Clientes c
JOIN
    Pedidos p ON c.idCliente = p.idCliente
ORDER BY
    c.nomeCliente;
```

Below the query editor, there is a table viewer titled "Clientes(+)" showing the results of the query. The table has 7 columns: "idCliente", "nomeCliente", "cidade", "numeroPedido", and "dataPedido". The results are as follows:

Grade	idCliente	nomeCliente	cidade	numeroPedido	dataPedido
1	1	Ana	São Paulo	1	2024-01-10
2	2	Bruno	Rio de Janeiro	2	2024-01-12
3	3	Carlos	São Paulo	3	2024-02-15

## Exemplo 3: Diferença

```
<localhost> Script  *<localhost> Script-1 x Clientes Sis
#use SistemaExemplo;
SELECT
    c.idCliente,
    c.nomeCliente,
    c.cidade
FROM
    Clientes c
LEFT JOIN
    Pedidos p ON c.idCliente = p.idCliente
WHERE
    p.idCliente IS NULL
ORDER BY
    c.nomeCliente;
```

Clientes 1 X				
<T SELECT c.idCliente, c.nomeCliente, c.cidade FR  Insira uma expressão				
Grade	123 idCliente	A-Z nomeCliente	A-Z cidade	
1	4	Daniel	Belo Horizonte	

# Exemplo 4: Produto Cartesiano

<localhost> Script

\*<localhost> Script-1 X

Cientes

SistemaExemplo

Compra

#use SistemaExemplo;

SELECT \* FROM Cientes, Produtos;

Clientes(+)		SELECT * FROM Cientes, Produtos						
		Insira uma expressão SQL para filtrar os resultados (use Ctrl+Espaço)						
Grade		idCliente	nomeCliente	cidade	idProduto	nomeProduto	categoria	precoProduto
1		1	Ana	São Paulo	1	Celular	Eletrônicos	1.500
2		2	Bruno	Rio de Janeiro	1	Celular	Eletrônicos	1.500
3		3	Carlos	São Paulo	1	Celular	Eletrônicos	1.500
4		4	Daniel	Belo Horizonte	1	Celular	Eletrônicos	1.500
5		1	Ana	São Paulo	2	Notebook	Eletrônicos	3.500
6		2	Bruno	Rio de Janeiro	2	Notebook	Eletrônicos	3.500
7		3	Carlos	São Paulo	2	Notebook	Eletrônicos	3.500
8		4	Daniel	Belo Horizonte	2	Notebook	Eletrônicos	3.500
9		1	Ana	São Paulo	3	Mesa	Móveis	800
10		2	Bruno	Rio de Janeiro	3	Mesa	Móveis	800
11		3	Carlos	São Paulo	3	Mesa	Móveis	800
12		4	Daniel	Belo Horizonte	3	Mesa	Móveis	800
13		1	Ana	São Paulo	4	Cadeira	Móveis	300
14		2	Bruno	Rio de Janeiro	4	Cadeira	Móveis	300
15		3	Carlos	São Paulo	4	Cadeira	Móveis	300
16		4	Daniel	Belo Horizonte	4	Cadeira	Móveis	300

# Exemplo 5: Restrição

<localhost> Script

\*<localhost> Script-1 x

Clientes

SistemaExemplo

Compras

#use SistemaExemplo;

SELECT \* FROM Clientes WHERE cidade = 'São Paulo';

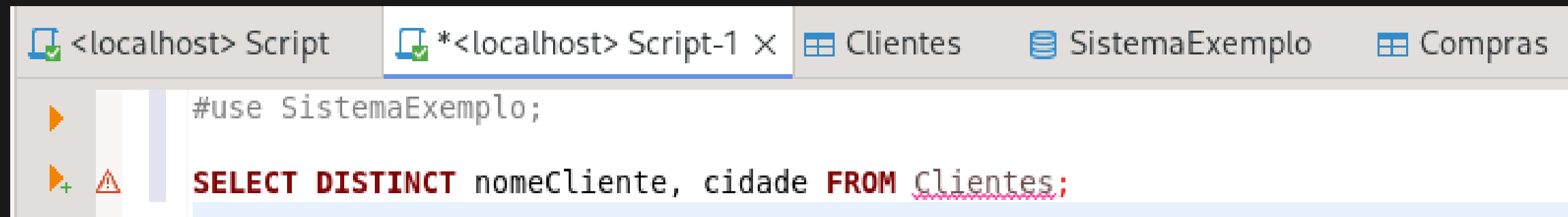
Clientes 1 x

SELECT \* FROM Clientes WHERE cidade = 'São

Insira uma expressão SQL para filtrar os resultados (us

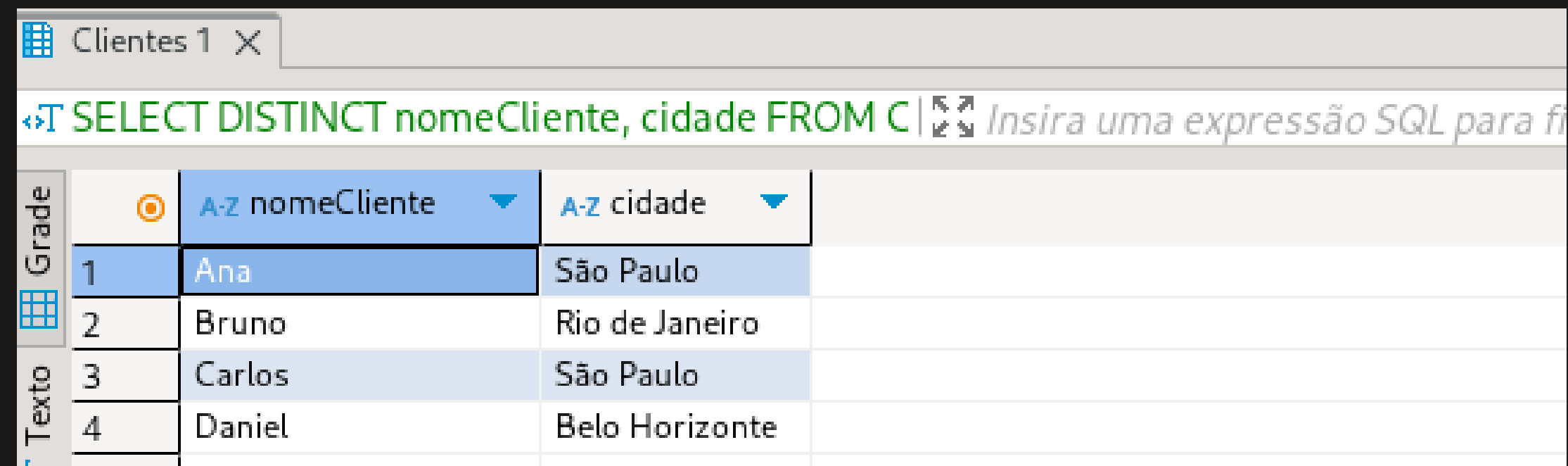
	<div>123 idCliente</div>	<div>A-Z nomeCliente</div>	<div>A-Z cidade</div>	
1	1	Ana	São Paulo	
2	3	Carlos	São Paulo	

## Exemplo 6: Projeção



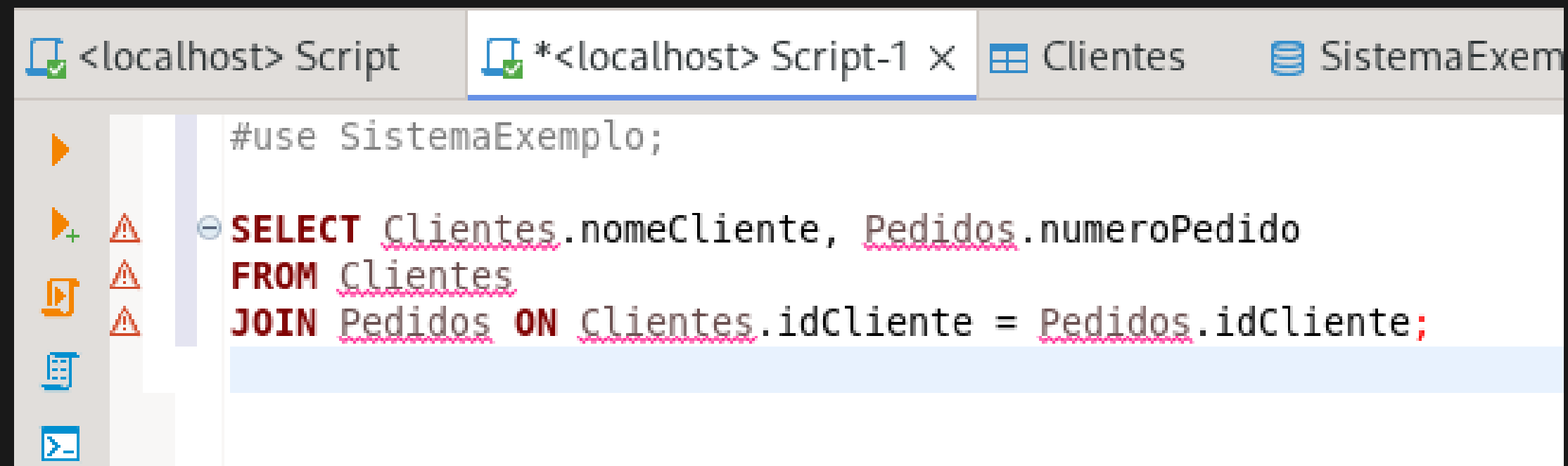
```
#use SistemaExemplo;

SELECT DISTINCT nomeCliente, cidade FROM Clientes;
```



	A-Z nomeCliente	A-Z cidade
1	Ana	São Paulo
2	Bruno	Rio de Janeiro
3	Carlos	São Paulo
4	Daniel	Belo Horizonte

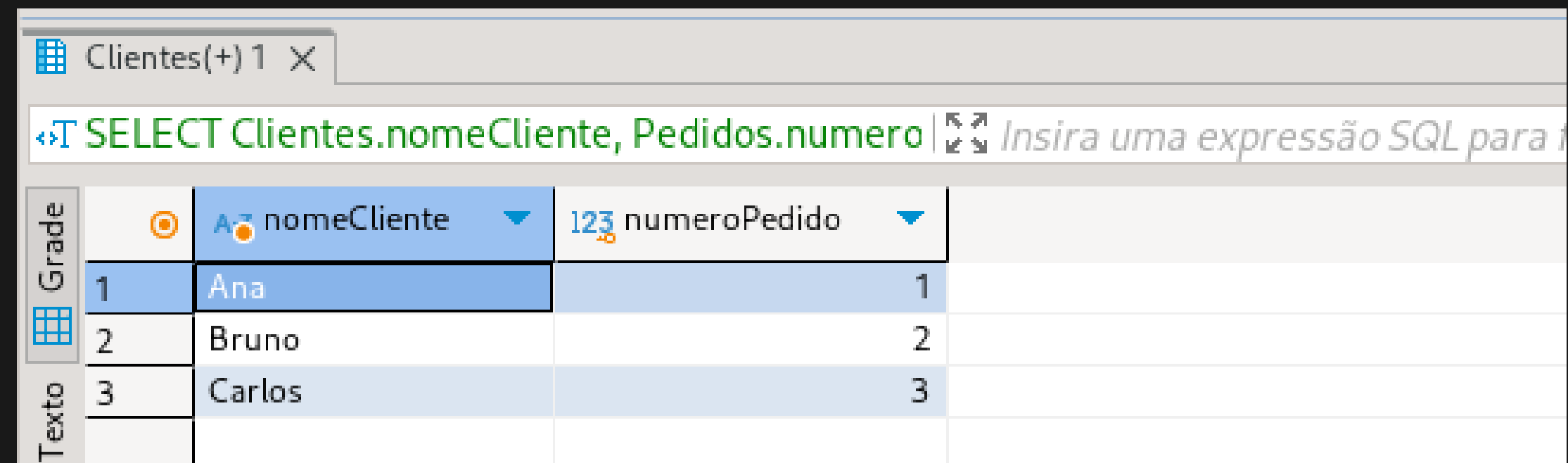
## Exemplo 7: Junção



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a script window titled "<localhost> Script-1" containing the following SQL query:

```
#use SistemaExemplo;  
  
SELECT Clientes.nomeCliente, Pedidos.numeroPedido  
FROM Clientes  
JOIN Pedidos ON Clientes.idCliente = Pedidos.idCliente;
```

The bottom pane shows the "Clientes" table structure and data.



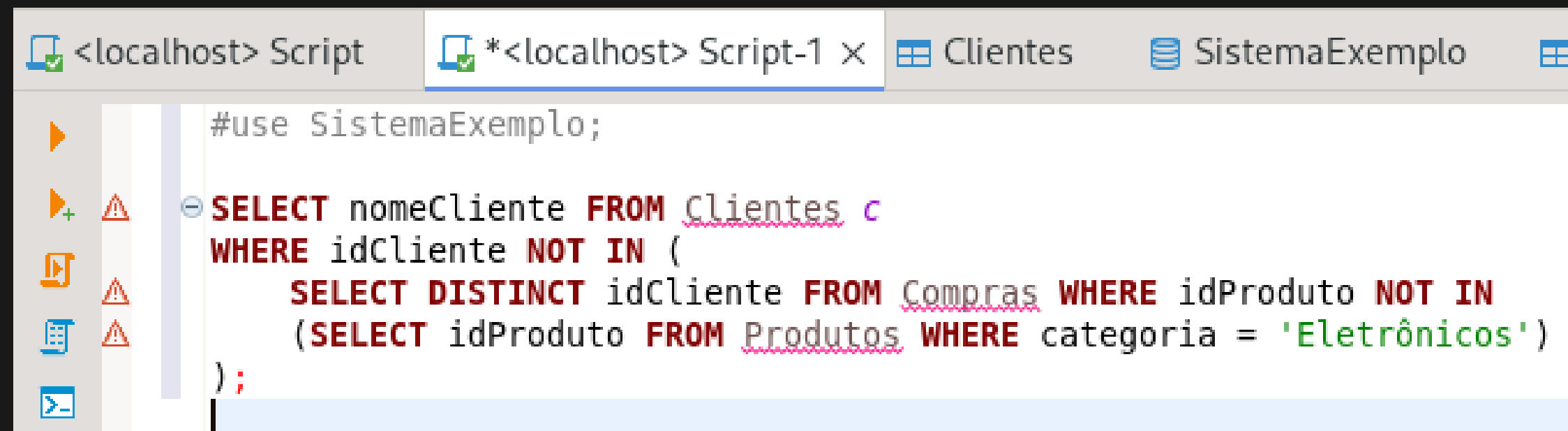
The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a script window titled "<localhost> Script-1" containing the following SQL query:

```
#use SistemaExemplo;  
  
SELECT Clientes.nomeCliente, Pedidos.numeroPedido  
FROM Clientes  
JOIN Pedidos ON Clientes.idCliente = Pedidos.idCliente;
```

The bottom pane shows the "Clientes" table structure and data.

Grade	nomeCliente	numeroPedido
1	Ana	1
2	Bruno	2
3	Carlos	3

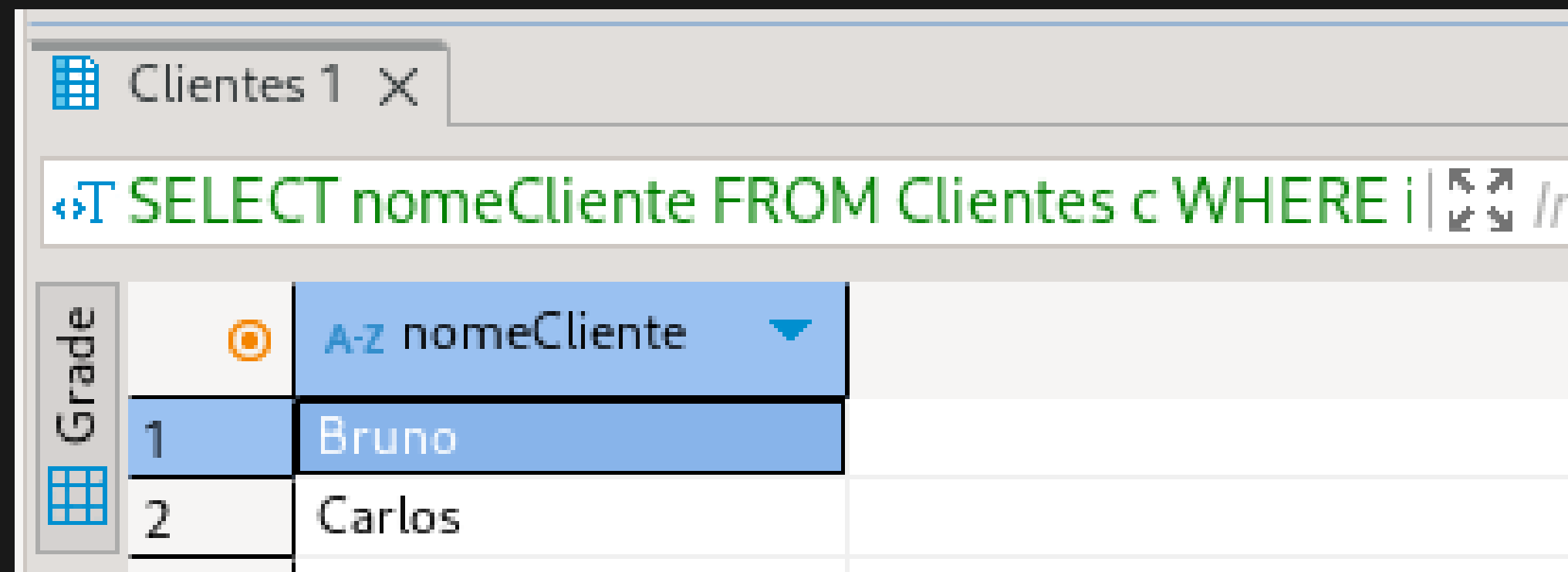
## Exemplo 8: Divisão



The screenshot shows the SQL Server Enterprise Manager interface. The 'Script' window is active, displaying a query that uses a subquery to filter clients based on product categories. The query is as follows:

```
#use SistemaExemplo;

SELECT nomeCliente FROM Clientes c
WHERE idCliente NOT IN (
    SELECT DISTINCT idCliente FROM Compras WHERE idProduto NOT IN
    (SELECT idProduto FROM Produtos WHERE categoria = 'Eletrônicos')
);
```



The screenshot shows the 'Clientes 1' window displaying the results of the query. The results are shown in a table view with a 'Grade' column and a 'nomeCliente' column. The table contains two rows of data.

Grade	nomeCliente
1	Bruno
2	Carlos



# Para que serve a álgebra relacional?

- Álgebra Relacional: Conjunto de operações (união, interseção, diferença, produto, restrição, projeção, junção, divisão, renomeação de atributos) usado para manipulação de dados.
- Principais Aplicações:
  - a. Busca de Dados: Definir escopos para consultas e atualizações.
  - b. Definição de Restrições: Integridade, segurança, estabilidade e RelVars derivadas.
  - c. Otimização de Consultas: Manipulação simbólica para melhorar eficiência.
  - d. Medida de Poder Expressivo: Avaliar completude relacional de linguagens.





# Pontos avançados

# Associatividade e Comutatividade dos Operadores

- Operadores Associativos:
- UNION, INTERSECT, TIMES, JOIN
  - Exemplo:
  - $(a \text{ UNION } b) \text{ UNION } c \equiv a \text{ UNION } (b \text{ UNION } c) \equiv a \text{ UNION } b \text{ UNION } c$
- Operadores Comutativos:
- UNION, INTERSECT, TIMES, JOIN
  - Exemplo:
  - $a \text{ UNION } b \equiv b \text{ UNION } a$
- Exceção:
- MINUS não é associativo nem comutativo.

# Equivalências Importantes

---

- Restrições:
  - $r \text{ WHERE TRUE} \equiv r$
  - $r \text{ WHERE FALSE} \equiv \text{vazio}$
- Projeções:
  - $r \{X, Y, \dots, Z\} \equiv r$  (se  $X, Y, Z$  forem todos os atributos)
  - $r \{\} \equiv \text{TABLE\_DEE}$  (se  $r$  não for vazio)
- Operações:
  - $r \text{ JOIN } r \equiv r \text{ UNION } r \equiv r \text{ INTERSECT } r \equiv r$
  - $r \text{ UNION vazio} \equiv r$
  - $\text{vazio INTERSECT } r \equiv \text{vazio}$

# Generalizações dos Operadores

---

- Generalização para  $n$  operandos:
  - JOIN, UNION e INTERSECT podem ser aplicados a mais de dois operandos ( $n$ -ádicos).
- Caso  $n = 1$ :
  - A junção, união ou interseção de uma única relação é a própria relação.
- Caso  $n = 0$ :
  - JOIN: TABLE\_DEE
  - UNION: Relação vazia do mesmo tipo.
  - INTERSECT: Relação universal (contém todas as tuplas possíveis).

# Operadores adicionais



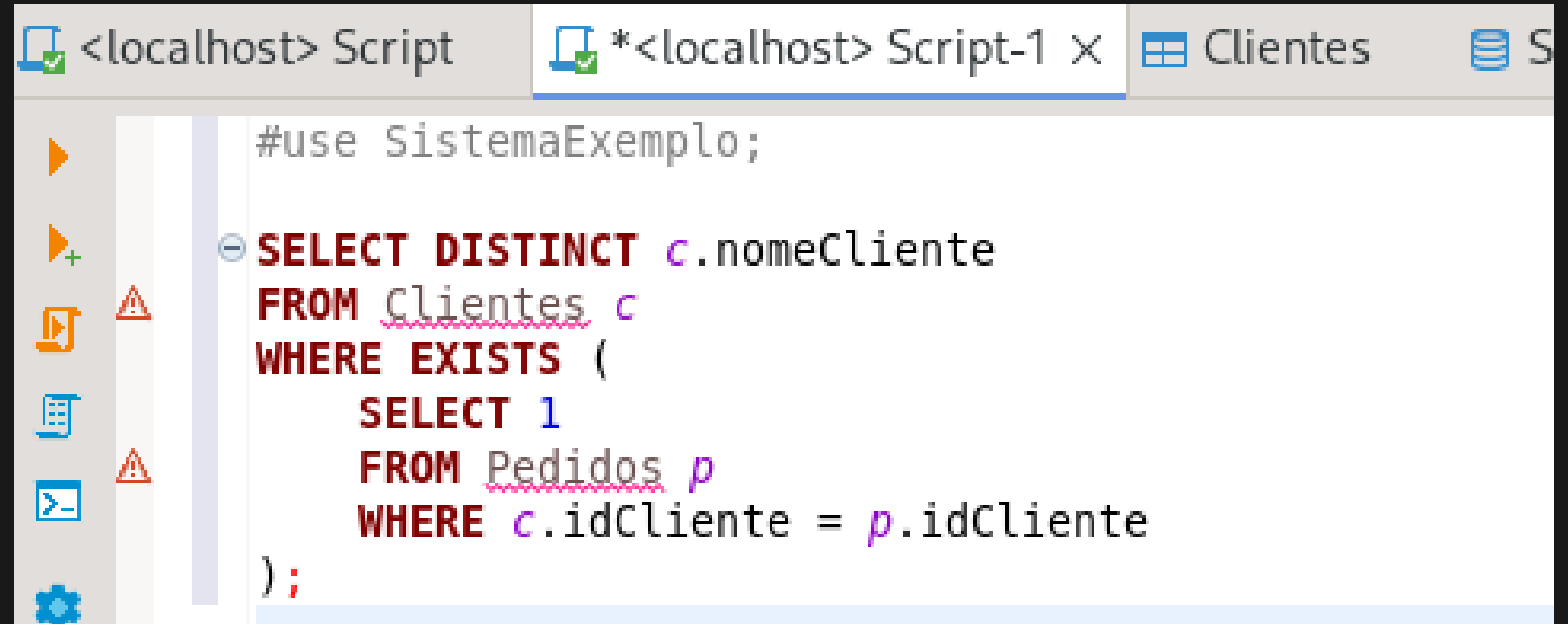
**Operações de junção e  
diferença no contexto  
de bancos de dados  
relacionais**

**Diferença entre junção  
completa e operações  
semi (semijunção e  
semidiferença)**

**Aplicação em  
consultas SQL para  
filtragem de dados**

# Semijunção (SEMIJOIN)

**Retorna as tuplas de uma relação (A) que têm correspondências em outra relação (B), sem incluir colunas de B.**



```
#use SistemaExemplo;

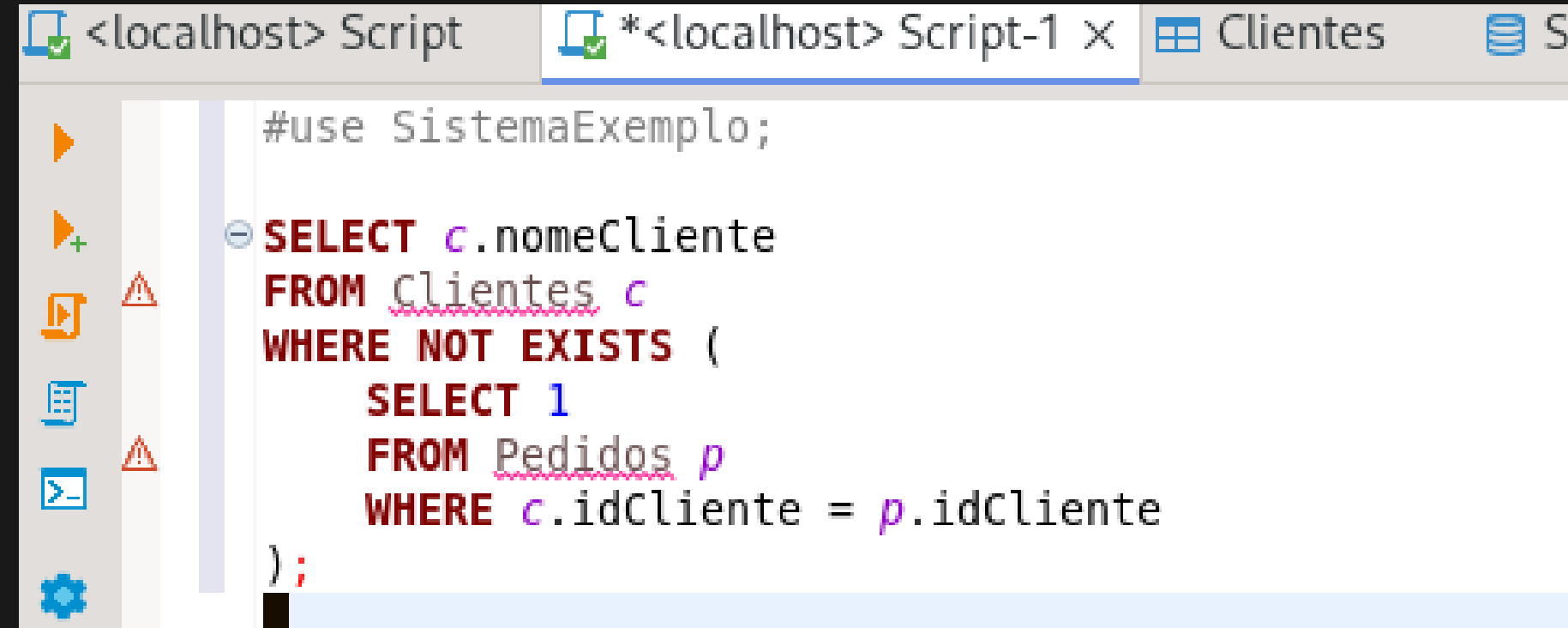
SELECT DISTINCT c.nomeCliente
FROM Clientes c
WHERE EXISTS (
    SELECT 1
    FROM Pedidos p
    WHERE c.idCliente = p.idCliente
);
```

The screenshot shows a SQL script editor with a toolbar on the left containing icons for running, saving, and other operations. The script is written in a standard SQL dialect and uses a semi-join to find clients with orders. The query is highlighted with a blue selection bar.

**Listar clientes que têm pedidos, mas sem detalhar os pedidos.**

# Semidiferença (SEMIMINUS)

**Definição:** Retorna as tuplas de uma relação (A) que não têm correspondências em outra relação (B).

A screenshot of a SQL script editor window. The window has three tabs: '<localhost> Script', '\*<localhost> Script-1 x', and 'Clientes'. The 'Script-1' tab is active. The script content is as follows:

```
#use SistemaExemplo;  
  
SELECT c.nomeCliente  
FROM Clientes c  
WHERE NOT EXISTS (  
    SELECT 1  
    FROM Pedidos p  
    WHERE c.idCliente = p.idCliente  
);
```

The script is written in a dark-themed editor with syntax highlighting. The 'Clientes' and 'Pedidos' tables are underlined in the FROM clause. There are two warning icons (yellow triangles) in the left margin of the script editor.

**Encontrar clientes que não fizeram pedidos.**

# Extensão (EXTEND)

Adiciona colunas a uma relação com valores derivados de cálculos ou expressões.

```
<localhost> Script  *<localhost> Script-1 x  Clientes  SistemaExemplo  Comp
#use SistemaExemplo;
SELECT nomeProduto, precoProduto, precoProduto * 0.90 AS precoComDesconto
FROM Produtos;
```

	A-Z nomeProduto ▼	123 precoProduto ▼	123 precoComDesconto ▼
1	Celular	1.500	1.350
2	Notebook	3.500	3.150
3	Mesa	800	720
4	Cadeira	300	270

Criar uma coluna calculada que aplica um desconto ao preço dos produtos.



# Totalização (SUMARIZE)

Agrega dados em uma relação, aplicando funções como soma, média, contagem, etc.

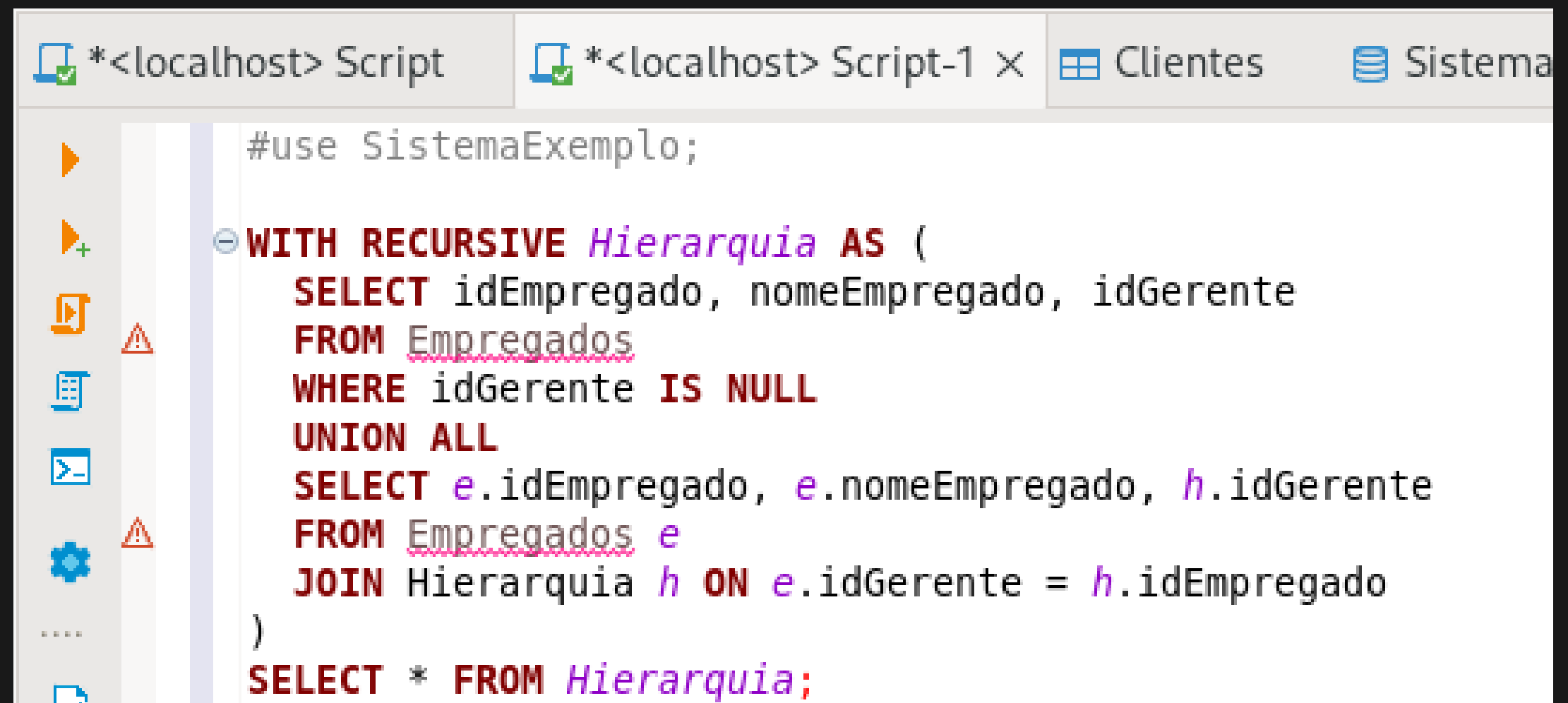
```
<localhost> Script  *<localhost> Script-1 x  Clientes  Sister
#use SistemaExemplo;
SELECT idCliente, SUM(valorCompra) AS totalCompras
FROM Compras
GROUP BY idCliente;
```

Grade	123 idCliente	123 totalCompras
1	1	2.300
2	2	1.500
3	3	3.500
4	4	300

Calcular o valor total das compras de cada cliente.

# Tclose (Fecho transitivo)

Encontra todas as conexões diretas e indiretas em uma relação (usado para hierarquias).



```
*<localhost> Script  *<localhost> Script-1 x  Clientes  SistemaExemplo

#use SistemaExemplo;

WITH RECURSIVE Hierarquia AS (
    SELECT idEmpregado, nomeEmpregado, idGerente
    FROM Empregados
    WHERE idGerente IS NULL
    UNION ALL
    SELECT e.idEmpregado, e.nomeEmpregado, h.idGerente
    FROM Empregados e
    JOIN Hierarquia h ON e.idGerente = h.idEmpregado
)
SELECT * FROM Hierarquia;
```

Listar todos os subordinados diretos e indiretos de um gerente usando CTE recursiva.

# Agrupamento e Desagrupamento

**Agrupamento e desagrupamento são operações que permitem mapear entre relações que contêm atributos com valores de relação e aquelas que não os contêm.**

**Essas operações são úteis quando lidamos com dados aninhados, em que um atributo pode conter várias tuplas.**

**Exemplos de uso prático incluem a agregação de dados relacionados e a sua subsequente separação.**



# Agrupamento (GROUP)

- Agrupamento cria uma nova relação na qual alguns atributos são combinados em um único valor de relação.

- Exemplo de expressão:

```
FP GROUP { P#, QDE } AS PQ
```

- Agrupamento por F#: Para cada valor distinto de F# na tabela FP, os atributos P# e QDE são agrupados em uma relação.
- Estrutura Resultante
  - Cabeçalho: { F#, PQ (com atributos P# e QDE) }
  - Corpo: Uma tupla para cada valor distinto de F#.

F#	PQ	
F1	P#	QDE
	P1	300
	P2	200
	P3	400
	P4	200
	P5	100
	P6	100
F2	P#	QDE
	P1	300
	P2	200
F3	P#	QDE
	P2	200
F4	P#	QDE
	P2	200
	P4	300
	P5	400

# Desagrupamento (UNGROUP)

- O desagrupamento desfaz o agrupamento, separando os atributos de uma relação aninhada em atributos individuais.
- Exemplo de expressão:  

```
FPQ UNGROUP PQ
```
- Desagrupamento de PQ: A relação FPQ é convertida de volta à relação FP original, com os atributos P# e QDE separados.
- Estrutura Resultante
  - Cabeçalho: { F#, P#, QDE }
  - Corpo: Uma tupla para cada combinação de F# e sua relação associada de P# e QDE.

DOIS	A	RVX			
	1	<table><tr><td>X</td></tr><tr><td>a</td></tr><tr><td>b</td></tr></table>	X	a	b
X					
a					
b					
	1	<table><tr><td>X</td></tr><tr><td>a</td></tr><tr><td>c</td></tr></table>	X	a	c
X					
a					
c					

TRÊS

A	X
1	a
1	b
1	c

UM	A	RVX			
	1	<table><tr><td>X</td></tr><tr><td>a</td></tr><tr><td>b</td></tr><tr><td>c</td></tr></table>	X	a	b
X					
a					
b					
c					



**RESUMO**



# Introdução à Álgebra Relacional

- Importância do Fechamento:  
Regras de inferência e expressões relacionais aninhadas.
- Operadores Primitivos:  
Conjunto inicial de operadores que formam a base da álgebra relacional.

# Operadores da Álgebra Relacional

- Operadores Tradicionais:  
União, Interseção, Diferença, Produto.
- Operadores Relacionais Especiais:  
Restrição, Projeção, Junção, Divisão (substituível por comparações).

# Novos Operadores e Conceitos

- Novos Operadores:  
RENAME, SEMIJOIN, SEMIMINUS, EXTEND, SUMMARIZE, GROUP, UNGROUP.
- Abstração e Eficiência:
  - Definição de operadores em termos de primitivos.
  - Uso de WITH para simplificação de expressões complexas.



# EXERCÍCIOS

# EX 1:

- Liste os nomes dos clientes que moram em cidades diferentes de São Paulo.

## EX 2:

- Liste todos os produtos disponíveis combinados com todos os clientes que moram em Belo Horizonte.

## EX 3:

- Encontre a quantidade de produtos comprados por cada cliente.

## EX 4:

- Liste os nomes dos produtos e suas categorias, mas renomeie as colunas para "Produto" e "Tipo".



# EX 5:

- Liste os clientes que não têm compras registradas.



# Isso é tudo pessoal!

"O futuro tem muitos nomes. Para os fracos, é o inatingível. Para os temerosos, o desconhecido. Para os valentes, é a oportunidade."

Victor Hugo em Os Miseráveis:





# Referências bibliográficas

---

DATE, C. J.; ANDRÉ VIEIRA; LIFSCHITZ, S. Introdução a sistemas de bancos de dados. Rio De Janeiro: Campus, 2004.

KORTH, H. F.; SILBERSCHATZ, A. Sistema de banco de dados. [s.l: s.n.].